```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
#
#The MIT License
#
#Copyright (c) 2011
#
#Permission is hereby granted, free of charge, to any person obtaining a copy
#of this software and associated documentation files (the "Software"), to deal
#in the Software without restriction, including without limitation the rights
#to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
#copies of the Software, and to permit persons to whom the Software is
#furnished to do so, subject to the following conditions:
#
#The above copyright notice and this permission notice shall be included in
#all copies or substantial portions of the Software.
#
#THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
#IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
#FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
#AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
#LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
#OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
#THE SOFTWARE.
#
#Authors:
#   Vili Auvinen (vili.k.auvinen@jyu.fi)
#   Olli Kauppinen (olli.kauppinen@jyu.fi)
#   Juho Tammela (juho.i.tammela@jyu.fi)
'''The module contains the classes for reading and storing the data of a requirements XML file

Requirements object contains a list of Requirement objects.

@author: Vili Auvinen, Olli Kauppinen, Juho Tammela
'''

class Requirements:
    '''The class is used for reading a requirements file and storing it in a list of Requireme

    @param requirementsFile: The requirements file as a DOM-tree.
    '''
    def __init__(self, requirementsFile):

        self.requirements = self.getRequirements(requirementsFile)


    def getRequirements(self, requirementsFile):
        '''Loops the requirement elements and stores them in a list as Requirement objects.

        @return: The list of Requirement objects.
        '''
        reqs = []
        for requirement in requirementsFile.getElementsByTagName('requirement'):
            requirementObject = Requirement(requirement)
            reqs.append(requirementObject)

        return reqs


class Requirement:
    '''The class is used for storing the information of one requirement.

    @param requirement: The requirement as a DOM element.
    '''
```

```
    def __init__(self, requirement):
        # Use the requirement name to guide it to the correct word_processing function.
        self.name = requirement.getAttribute('name')
        self.category = requirement.getAttribute('category')
        self.expectedValue = self.getExpectedValues(requirement)
        self.feedback = self.getFeedbacks(requirement)

    def getErrorMessage(self, errorValue = "DEFAULT"):
        '''Gets a error message from the requirement.

        @param errorValue: The errorvalue of the error message, defaults to 'DEFAULT'.
        '''
        #errorMessage = ""
        try:
            message = self.feedback[errorValue]
            return message
        except:
            return self.feedback["DEFAULT"]

    def getExpectedValues(self, requirement):
        '''Gets all the expected values of a requirement.

        @param requirement: The requirement as a DOM element.
        '''
        expectedElements = requirement.getElementsByTagName('expectedvalue')
        if len(expectedElements) == 1:
            return expectedElements[0].firstChild.nodeValue
            #return self._bool(expectedElements[0].firstChild.nodeValue)
            #return True

        properties = {}
        for valueElement in requirement.getElementsByTagName('expectedvalue'):
            properties[valueElement.getAttribute('name')] = valueElement.firstChild.nodeValue
        return properties

    def getFeedbacks(self, requirement):
        '''Gets all the error feedback messages.

        @param requirement: The requirement as a DOM tree.
        '''
        feedbacks = {}
        for feedback in requirement.getElementsByTagName('feedback'):
            errorvalue = feedback.getElementsByTagName('errorvalue')[0].firstChild.nodeValue
            text = feedback.getElementsByTagName('text')[0].firstChild.nodeValue
            feedbacks[errorvalue] = text
        return feedbacks


#    def _bool(self, s):
#        return s.upper() == 'TRUE'

#if __name__ == "__main__":
#    filename = 'sampleFiles/xml/requirements_test1.xml'
    #file = open(filename)

#    reqDom = xml.dom.minidom.parse(filename)
#    reqs = Requirements(reqDom)
    #print reqs.requirements
#    for req in reqs.requirements:
#        print req.name
        #print req.expectedValue
#        print req.expectedValue
#        print req.getErrorMessage()
```